A meta-model of CNC part programming languages

Xianzhi Zhang∗, Aydin Nassehi, Stephen T. Newman

Department of Mechanical Engineering, University of Bath, Bath, BA2 7AY, UK

∗Corresponding author:

Xianzhi Zhang

Email:xianzhi.zhang@bath.edu

Phone:+44(0)1225 384049

Fax:+44(0)1225 386928

A meta-model of CNC part programming languages

Abstract

Over the last 50 years the development of CNC machines has seen a plethora of part programming languages being developed. The large majority of these languages are based on the ISO 6983 standard which is commonly known as G&M codes, but other languages from machine tool suppliers are also used for programming machine tools. These programming languages have provided major barriers for the interoperability of such information between CNC machines and also from CNC machines to Computer-aided (CAx) systems. Thus the process knowledge in existing part programs cannot easily be recycled and reused, due to an inability to interpret these forms of data. In this paper, a new meta-model of CNC part programming languages has been proposed. The meta-model aims to abstract the characteristics of CNC machine activities and interpret/represent process information within CNC part programs written in different languages or dialects. Realising the translation between the meta-model and different part programming languages, it is possible to capture the shopfloor knowledge from CNC machines without the need to develop individual interpreting interfaces for each programming language. The valuable process knowledge can thus be captured from the part programs and represented in a neutral presentation to facilitate the knowledge accumulation and management, which is vital for manufacturing companies to gain competitive advantages in the globalised market.

Key words: CNC, part programs, interoperability, meta-model; G&M codes;

1. Introduction

The electronic files used to control CNC machines are often in a format, informally called G-codes, after Gerber Scientific Instruments, a manufacturer of photoplotters and the developer of the file format (Schroeder 1998). Besides G-codes, there are usually some other commands starting with M to program some other Miscellaneous functions such as coolant on/off, chip removal. Hence, this kind of part program is also referred to as G&M codes. A file containing G&M codes would comprise many lines of text that would be interpreted as moving instructions for the servomechanisms connected to various axes of

the machines (Smid 2003). With the wide use of G&M codes, it became the de-facto standard for CNC programming and then was formalised as RS-274D (USA), ISO6983 (ISO) and DIN66025 (Europe) (Liu et al. 2007).

Compared with the first generation of Numerical Controlled (NC) machine, the modern CNC machines have got many advanced capabilities of such as multi-axis control, adaptive control, error compensation and multi-process manufacture (Nguyen and Stark 2005). With the versatility of CNC machines, the programming task becomes increasingly more difficult. For some precision and complex jobs, it is impractical to program at the shopfloor, which makes offline computer aided software tools a necessity for efficient generation and verification of CNC codes. These software packages are usually called Computer Aided Manufacturing (CAM) systems. CAM systems together with Computer Aided Design (CAD) systems and Computer Aided Process Planning (CAPP) systems make up a Computer Aided system (CAx) chain. Along the chain, an information flow is formed from CAx systems to CNC machines.

With the rapid development of different machines and computer technology, CNC technology has advanced dramatically supporting those newly emerged machine functions. However, CNC controller vendors are very protective of every advance they made and employ proprietary standards for the enhancements that they introduced in their new controllers (Nassehi 2007). For example, CNC controller manufacturers introduced non-standard G-codes into the ISO 6983 standard to support their new features (additional axes, special caned cycles) resulting in various dialects for different machine and controller combinations (Proctor et al. 2002). Another reason why different dialects or languages are used to control the machine tool, the authors believe, is the simultaneous development of CNC machines tools around the world. People use different commands to program the same function. The standard (ISO6983-1 1982) of programming CNC machines actually came many years later after the first NC machine was launched and the standard is proposed based industrial practices.

Since different CNC machines use unique forms of part program, postprocessors have been used to generate the correct part programs for various machines. CAM vendors have to make effort to provide comprehensive postprocessors for each machine-controller configuration to be able to generate the correct dialect of G-codes for that specific combination (Hardwick and Loffredo 2006). As shown in Figure 1, each CAM system provides a unique postprocessor for every machine-controller combination. Since there are many different CAM systems which they employ proprietary data and algorithms, the

demand for postprocessors is huge. It should be noted that the controllers of different versions are treated as different controllers even with the same brand. GibbsCAM (GibbsCAM 2012) claimed that there were more than 10000 postprocessors in their library. The number of postprocessors also indicates the number of programming dialects/languages. Due to different languages and combinations of controllers and machine designs, it is extremely difficult to reuse a part program on a different machine.

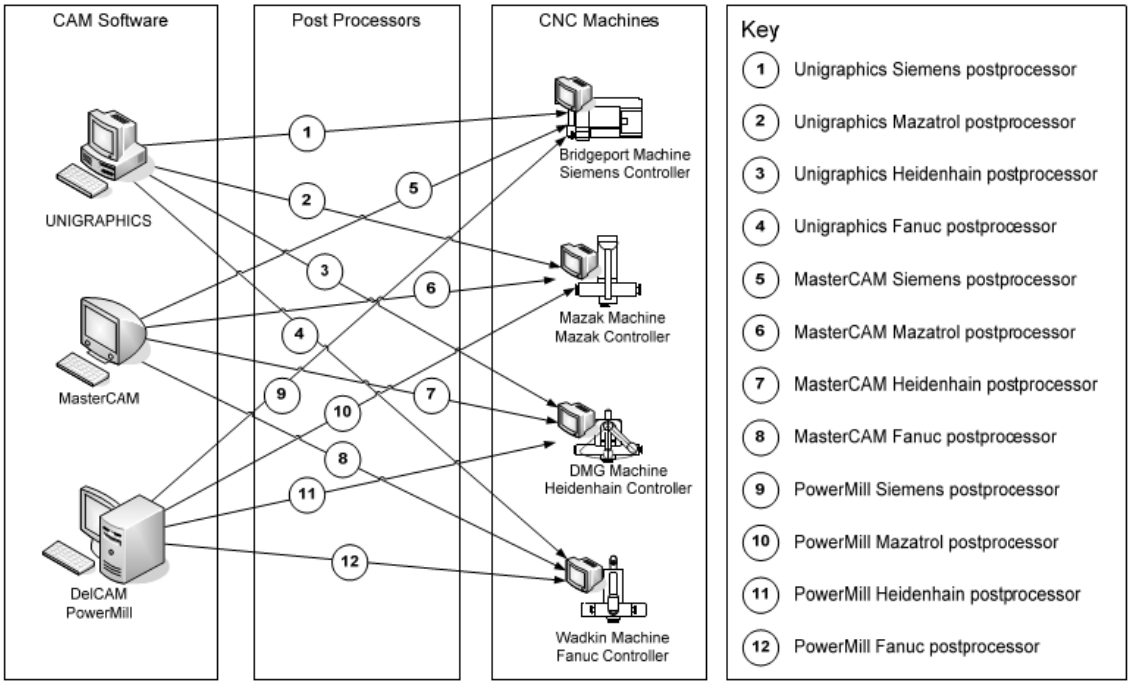


Figure 1 – Example of post processors for different CAM and CNC machines combinations

The problem in terms of information interoperability in the current CAx manufacturing chain is the information island of shopfloor. The information flow from CAx to the CNC machine is unidirectional and it is not usually possible to feedback the shopfloor information up the CAx chain. The information communication with the shopfloor is an essential part for the information interoperability of manufacturing. The fact of thousands of different programming dialects has prevented it from happening, especially in a uniform and automatic way (Newman *et al.* 2008). Hence, the CNC machine is actually an information island at the end of the manufacturing chain. However, the shopfloor is an important and knowledge intensive stage of production where valuable process knowledge or know-how is utilised by the manufacturing engineers (Zhang *et al.* 2013b). Since the shopfloor is a complex and ever-changing environment, and the part programs are generated based on a non-real-time resource status, the shopfloor engineers are entitled to make last-minutes changes to the part programs. From this point of view, the part programs are the last and most accurate

process plan used to machine the products. This onsite knowledge buried in thousands of part programs are difficulty to recycle and reuse.

Researchers (Liu et al. 2007; Schroeder and Hoffmann 2006) have tried to translate part programs and reuse them on different machines. The disadvantage of these solutions is that they only focuse on interoperability between CNC machines, not between the CNC machine and the CAx chain. Direct translation of dialectic part programs is not an effective way since there are too many different part programming languages. It works only under the assumption that the two different machines involved in exchanging part programs have the same physical axis configuration and use cutting tools with the same diameters and cutting heights. Going beyond any of these assumptions would require the toolpath and the part program to be generated again in the CAD/CAM system (Zhang *et al.* 2013b).

In this paper, a meta-model of different CNC programming languages has been proposed to represent the information and knowledge contained in CNC part programs. The characteristics of different programming languages have been analysed. An XML based translation method has been presented. Based on the meta-model, a Universal Process Comprehension interface (UPCi) has been developed to interpret the information in different part programs and represented it in a STEP-NC format. The implementation of the reverse information flow from shopfloor part programming to a STEP-NC based process plan has been termed process comprehension by the authors. The method using the meta-model to realise the interoperability between CNC machines and other systems (CAM or CNC) has been proven to be an effective solution through the case studies.

2. Comparison of part programming systems on modern CNC machines

Despite the different formats of part programming languages, from the semantics point of view, they are the same in essence. They all are motion commands for physical machine tools, which can be equipped with a controller of any make. That is the reason why identical or similar machine tools, such as 3-axis vertical CNC milling machines, can be controlled by either Fanuc or Siemens controllers. For the same manufactured part, identical machine tools equipped with different controllers perform similar or even identical motions with different part programs to machine the part. Thus, the real difference between programming languages is their presentations and interpreting methods. The logic and process knowledge contained in these part programs are the same. This can be substantiated from the mechanism of the CAM system. In a CAM system, after the process planner defines the

process plan, the micro-process data including tool path and associated switching operations such as tool change, coolant on/off etc. is calculated and generated. The next step is to choose the right postprocessor to convert this data to the part program for the corresponding CNC machine, as shown in Figure 2. The process plan (cutter location data, machine functions etc.) in the CAM system is resource independent, and theoretically, can be converted to any part program provided there is a suitable postprocessor available and the controller is capable to perform the task. A part program that is post processed becomes resource dependent; in other words, it will be tied to a specific combination of machine tool and controller as defined in the postprocessor. From this point of view, part programs are a machine specific representation of the underlying process plan. Theoretically, it is possible to convert different part programs back to a resource independent representation.

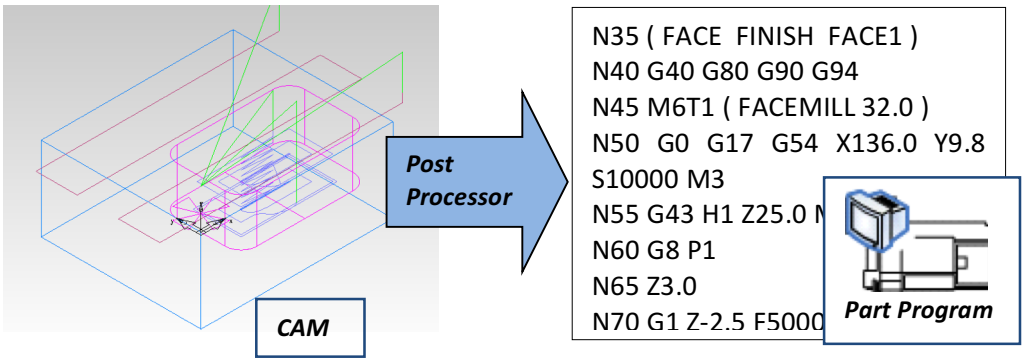


Figure 2 - Process plan in CAM to resource dependent part programs

From the syntax presentation of the programming languages, it is possible to translate the part program into a neutral language. For example in Table 1 (GE Fanuc Automation 1998; Siemens AG 2000; Heidenhain 2009), there are several commands formatted for a Fanuc controller, a Siemens controller and a Heidenhain controller. For linear interpolation (straight cutting travel of the cutting tool from one position to another specified by the coordinates), Fanuc uses G1 and Heidenhain uses L. Similarly, for clockwise circular interpolation commands in the XY-plane, Fanuc has two types of commands format by specifying the centre of the arc and the radius. Whilst the Siemens controller has two more command formats to identify the same arc toolpath: by specifying the opening angle or an intermediate point. Heidenhain controllers can be programmed using their proprietary language and have a special format for an arc path using command “CT” without defining centre and radius, as shown in Table 1. The arc path starts from last position at a tangent with the previously programmed contour element and ends with the current position. Although they use different command formats, the semantics behind them is the same. In this comparison shown in Table 1, they are ordering the cutting tool to move along a straight line, and an arc or drilling

a hole with chip breaking movements. Hence, if a neutral data model is available, different part programs can be translated into a neutral representation, provided a translation dictionary is available. In fact, there is such a neutral representation developed by NIST (Proctor *et al.* 1997) called Canonical Machining Commands to represent various programming dialects. The aim of the canonical commands is to represent RS274 (US equivalent of ISO 6983) codes to realise one-to-one correspondences with commands employed by commercial motion control boards (Liu *et al.* 2007; Guo *et al.* 2011). However, this model is designed for the RS274 language dialects and it embraces more specific and low-level data. For example, a peck drilling cycle in CNC commands can be decomposed into several, possibly dozens of, canonical commands. For this research, micro-process information contained in CNC part programs is already low-level information. There is no need to interpret this low-level information into a lower and more specific level. On the contrary, it is better to keep the information granularity to help to gain a higher-level understanding of the process data at the low shopfloor level. Consequently, a new meta-model of CNC programming languages is proposed in this research to represent the process information contained in part programs written in various programming dialects.

Table 1 - Comparisons between Fanuc, Siemens and Heidenhain programming dialects

Commands / Controller	Fanuc	Siemens	Heidenhain
Linear Interpolation	G01 X..Y..	G1 X..Y..	L X+..Y+..
Clockwise Circular Interpolation	G02 X..Y.. I..J..*or* G02 X..Y..R..	G2 X..Y..I..J.. *or* G2 X..Y..CR=..*or* G2 X..Y..AR=..*or* CIP X..Y..I1=..J1=..	C X+..Y+..DR..*or* CR X+..Y+..R+..*or* CT X+..Y+..
Peck Drilling Cycle	G83…	CYCLE 83…	Cycle 1…
Input Unit	G20 – Inch G21 – Metric	G70 – Inch G71 – Metric	INCH MM

Notes:

(i) Examples of NC commands in this table are applied for XY plane.
(ii) The values of the parameters are ignored and indicted by "..".
(iii) Peck Drilling cycle parameters are complex and not included in this table.

3. A meta-model of CNC programming languages

The aim of the meta-model is to unify the following interpreting or translation algorithms regardless the differences between programming dialects. As shown in Figure 3 (a), if there is not such a model, there is a need to develop separate interpreting interfaces for each type of CNC part programming dialects for different controllers (Fanuc 18i, Siemens 840D and Heidenhain iTNC 530). This work should be huge and not practical for implementation since there are thousands of programming dialects in existence (Maeder *et al.* 2002; STEP Tools Inc 2012). With this meta-model, different dialects can be translated and represented by this model and only one standardised interpreting interface (as shown in Figure 3(b)) is needed regardless of the type of the languages. It simplifies the task of the development of interpretation algorithms significantly. Also, the modularisation, to have separate modules of inputting and handling data, helps to make the system less complex and easy to test.

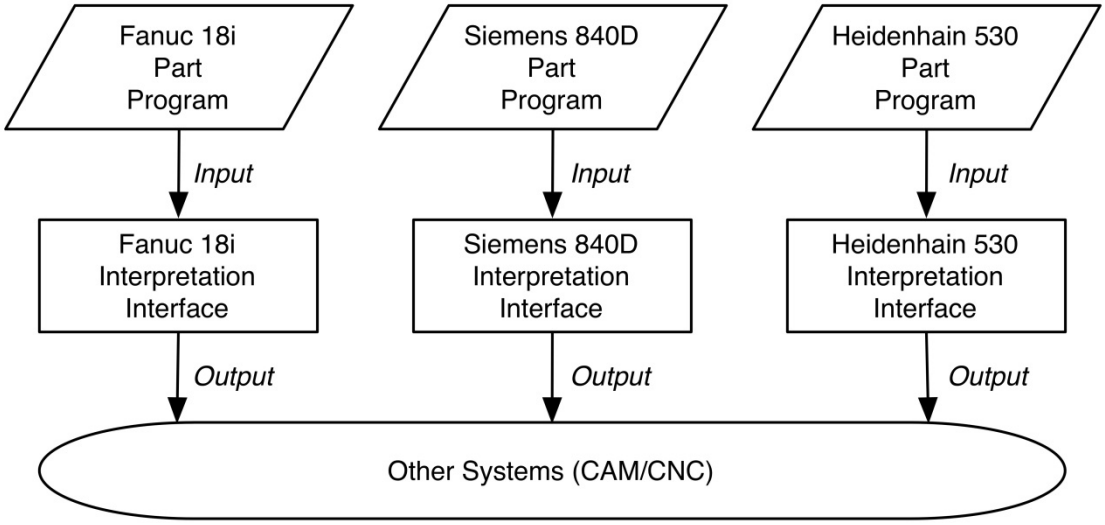
This meta-model has been devised with three objectives:

(i) The meta-model should cover general functionalities of common 3-axis CNC milling machine tools.
(ii) The meta-model has to be able to interpret part programs based on ISO 6983 commands and has mechanisms to incorporate other proprietary programming languages.
(iii) The meta-model should be able to represent the original process information and keep its integrity, granularity and homogeny.

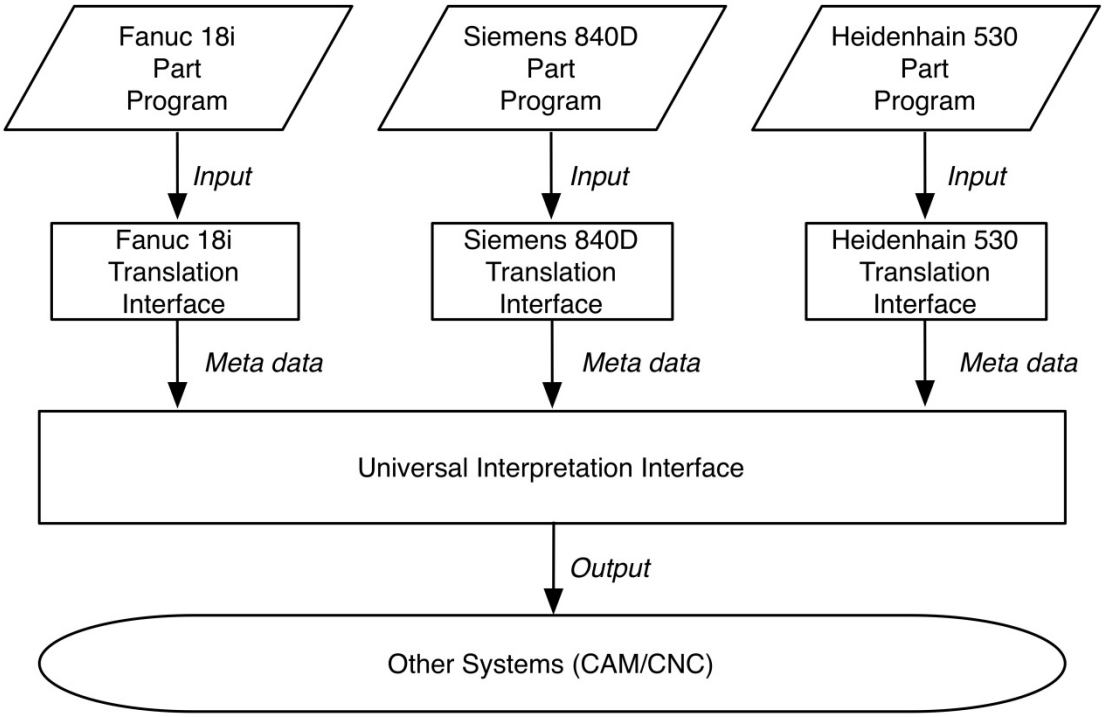
In this research a primary version of the meta-model has been developed. This version covers most of the motion commands and some miscellaneous settings for 3-axis machining. Four types of frequently used drilling cycles are included. A list of the meta-model of CNC activities is shown in Table 2.

The meta-model entities listed in Table 2 are developed based on the most widely used G&M codes to meet the three objectives specified above. Though it covers most of the common G&M commands, it is not restricted to G&M codes. Judging from the names of these entities and their descriptions, it covers common movements and settings of all kinds of CNC machines regardless of what kind of controllers are equipped on them. Hence it is possible to translate other CNC programming languages other than G&M based dialects into this model. Consequently it can represent the process information contained in part programs. Furthermore, it keeps and in some cases enhances the information granularity.

For example, there are separate commands to specify the plane selections (or measure units) in G&M codes. In the meta-model, there is only one entity to model these choices.



(a) Individual interpretation interfaces for each programing dialects



(b) Meta-model and universal interpretation interface for different programing dialects

Figure 3 - meta-model used for universal interpretation of part programs

Table 2 - meta-model entities of CNC activities

meta-model entities	***Functions***
RepidPosition	Repaid positioning.
LinearInterpolation	Linear movement of cutting tool.
CircularInterpolation	Circular movement of cutting tool.
Drilling	Drilling cycle or spot drilling cycle to create a common hole or a guide hole.
Boring	Boring cycle to enlarge an already existing hole using a single point cutter.
PeckDrilling	Multi-step drilling cycle to create a deep hole.
CounterBoring	Drilling cycle to create a stepped hole.
CycleCancel	Cancel canned cycle in modal (continuous effect).
CommandManner	Absolute or Incremental command.
PlaneSelection	Plane selection for circular interpolation.
Unit	Measure unit: MM/INCH.
ToolFunction	Tool call and tool change.
Feedrate	Feedrate.
Spindle	Spindle ON/OFF switch and spindle speed.
Coolant	Coolant ON/OFF.
End	Program stop/end.

4. Modelling NC languages in XML description

Although the meta-model simplifies the task of interpretation significantly since there is no need to develop different algorithms for each CNC dialect, there is a need to develop translation interfaces between CNC dialects and the meta-model. A traditional and straightforward way to do this is developing proprietary interfaces to read in different dialects, as shown in Figure 3 (b). It means there is a need to develop many of this type of interface in advance to support different dialects. In fact it is possible to use a “dictionary” mechanism to standardise these translation interfaces, as shown in Figure 4.

The dictionary mechanism uses a programming specification of CNC programming dialects in the form of XML to help the standardised interface to realise the translation from dialects

to the meta-model. The programming specification is a description of programming format and its correspondence representation or interpretation with the meta-model. An XML method is used to describe the programming specifications. In this research each XML file is an XML specification of a programming dialect. All of the XML based NC programming specifications assembly a dialects-meta dictionary, which works as a reference to assist the interface to translate CNC dialects into the meta-model. Each XML specification is an entry of the dictionary. Hence, it is possible to translate any CNC dialect through the interface by simply adding a new entry to the dialects-meta dictionary.

Using the dictionary method, only a single standardised translation interface is needed between CNC part program written in various dialects and the meta-model, as shown in Figure 5. Consequently, the translation interface and the meta-model can be encapsulated as a standalone system, as highlighted in Figure 5 in a dashed rectangular box. The system is titled Universal Process Comprehension interface (UPCi). The part program and the XML specification of the dialect can be treated as the input of the system. This structure of the system gives it the necessary robustness and expansibility.

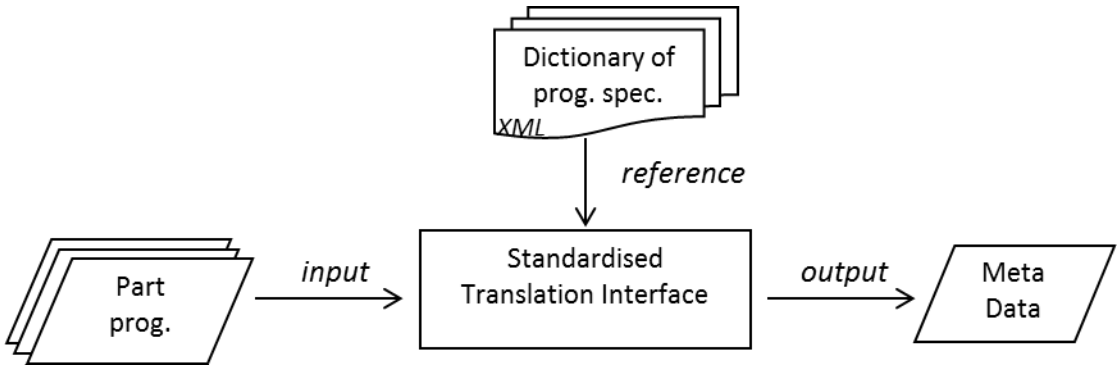


Figure 4 - Standardised translation interface using a “dictionary”

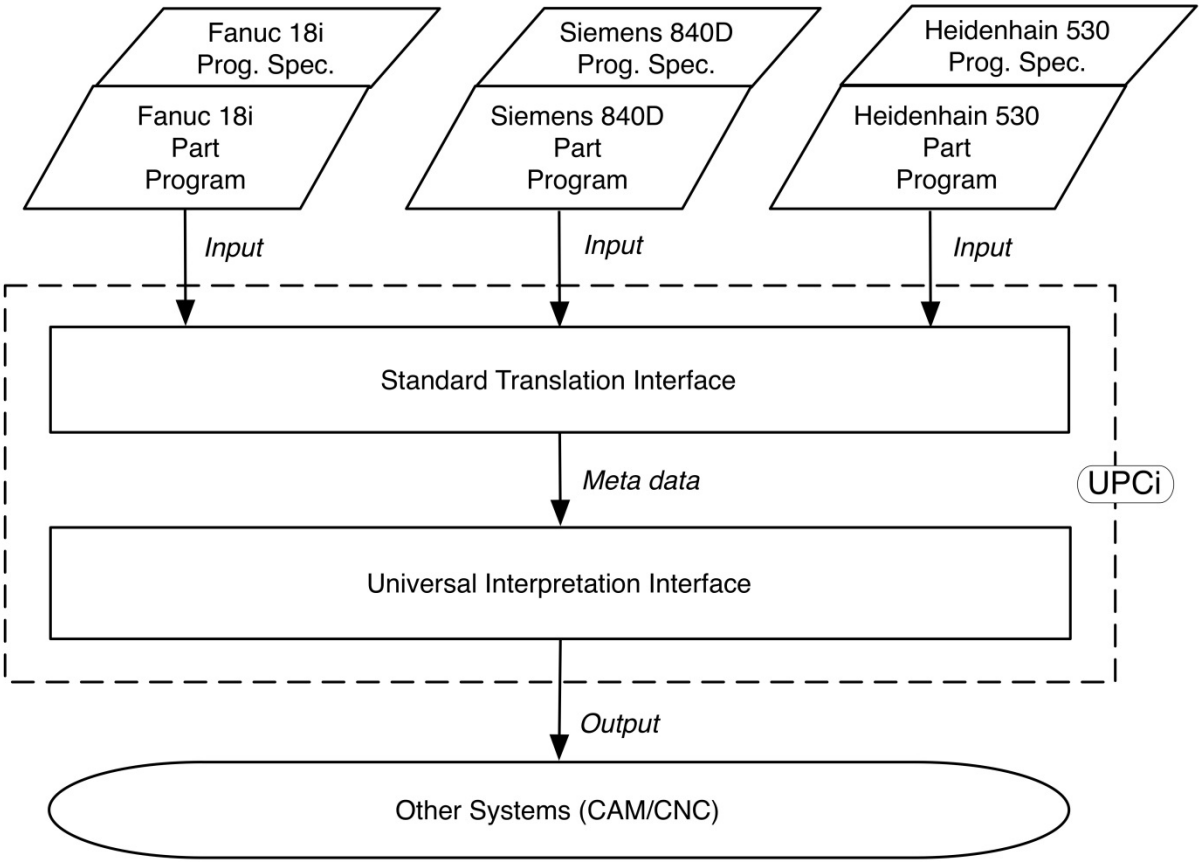


Figure 5 - Meta-model used for Universal Process Comprehension interface (UPCi)

In Figure 6, the XML specification has been designed for the Fanuc 18i controller. In this programming specification, the programming format for the controller is defined and mapped with the meta-model. The root tag is the beginning of the XML file. The first part of the file is the file header including the detail about the specification to imply: which controller this schema is applied with, when the schema is created and by who. After that is the basic grammar part: the basic grammar syntax about the axis words, comments etc. The third part of the XML file is the settings of the machine tool including the spindle, coolant, feedrate etc.

The forth part of the schema is about the motion commands under the *Movement* tag. The motion command illustrated in Figure 6 is the linear positioning of the cutting tool. The value of the *Modal* tag is *true*, which indicates this command is continually in effect without being explicitly programmed in the next command line, as shown in Figure 7. The line N180, N185 and N190 following the line N 175 mean the same linear-interpolation command and do not need to have “G1” explicitly programmed. The next tag is about the programming syntax for the command including the *Format*, *Parameters*. The *Format* tag defines the full presentation of the commands including all possible parameters. There can be more than one Format tag for a single command. For example, for Fanuc 18i controller, there are two format tags for clockwise circular interpolation to indicate the two options to program the cutting tool to

move in along a clockwise circular arc, as shown in Table 1. The parameters of each command are defined in the *Format* tag and started with dollar ($) symbols. The main command word(s) is embraced by two asterisks (*) on each side. In the *Parameters* tag, all of the parameters are listed with their functions. They are categorised into two groups: *Optional* and *Mandatory*, which indicate whether they are optional or compulsory for the command. For example in Figure 7, the position words *X*, *Y* or *Z* are optional for a *G0* rapid position commands, while the *Z* indicating the drilling depth and the *R* indicating the radius of the hole are mandatory for the *G83* drilling cycle commands.

Figure 6 - XML specification for Fanuc 18i controller

```
N175 G1 Z0.03 F1261.          N380 G0 G17 G54 X60.0 Y80.0 S1212 M3
N180 X54.57 Z-1.64            N385 G43 H3 Z25.0 M8
N185 X74.57 Z-3.31            N390 G83 G98 R3.0 Z-36.009 Q20.0F364.
N190 X54.57 Z-4.98            N395 G0 G80 Z25.0
```

Figure 7 - Part program segment for Fanuc 18i controller

The mapping between the CNC dialects and the meta-model is coupled with the help of NC programming specifications in the XML file. In the meta-model, for instance, there is a meta command of the linear interpolation ordering the cutting tool to cut along a straight line specified by the start and end coordinates. The Fanuc 18i uses G1 for the linear movements, which is described in the corresponding XML file, as shown in Figure 6. The translation interface of UPCi reads in the XML file and stores the description in memory. Then it reads in the part program and interprets it line by line. The detailed process to translate the part program to the meta-model is illustrated in Figure 8.

As shown in Figure 8, the process to interpret a part program starts from loading an appropriate XML specification. In the translation interface, there is an XML parser used to parse the XML file and store it in an XML parser object. Then the part program is loaded in and handled line by line. For each line, a pre-process operation is applied to format the line into a standard presentation, and get rid of the comments etc. The XML parser object provides the necessary information regarding syntax grammar, such as comments indicators, line number words. With the standard format, the XML parser object checks the syntax of that line: how many command keys in that line, all of the mandatory keys included or not, find out the keys and put them into a linear array. Then for each element in the array, a corresponding meta-model object is generated according to the keys listed within the XML parser object. For each meta-model object, the translation interface references the XML parser object to acknowledge the parameter words and sets values for the parameters of the meta-model object. After all of the command keys translated, the same process is repeated to the next line until the whole part program is processed and the translation activity finishes. The part program is then represented by the meta-model objects, which would be used as the input for process comprehension.

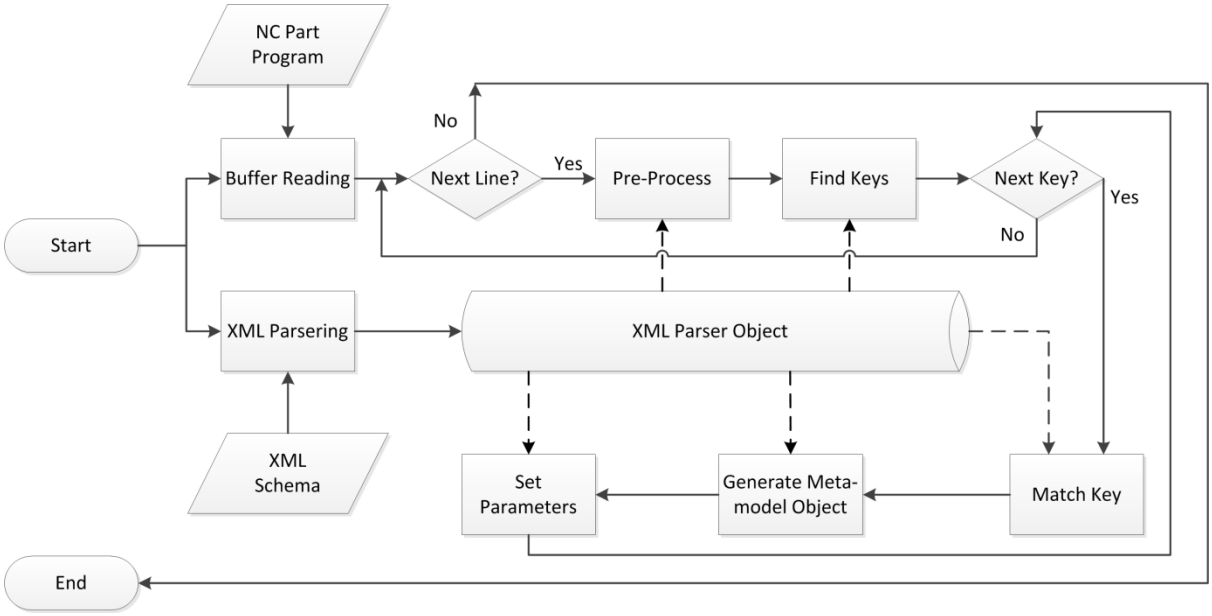


Figure 8 - Translation of part program to meta-model

The XML method used to describe the programming syntax enables UPCi to be expansible to support other programming languages. To achieve this, an XML file with the specification of the programming syntax is necessary. The users of UPCi can develop XML specifications according to their needs. In order to help the users to develop appropriate XML description files, a regulation is needed to make sure the XML files are in a uniform format to be suitable and understandable to UPCi. The regulation of XML file is used to define the legal building blocks of an XML specification of CNC programming languages.

There are many XML regulation methods available, of which the most widely used are Document Type Definition (DTD) and XML Schema or XML Schema Definition (XSD) (Lee and Chu 2012). Compared with DTD, XSD is more powerful and widely used (Bex *et al.* 2004). Moreover, XSD has built in data types and allows the user to custom data types (W3C 2012), which is convenient to this research. Thus, XSD has been chosen in this research to be used as the regulation method of XML specifications of CNC programming languages. The XML Schema used to regulate the XML description of CNC programming syntax has been developed as shown in Figure 9. In this schema, the basic blocks used to describe the syntax of CNC programming languages have been specified. Following the format, the user can develop XML specifications for other CNC programming languages. In implementation, the XML specifications will be checked against this schema to validate the format of the specifications.

```
Schema.xml | (no symbol selected)
<xs:element name="Make" type="xs:string"/>
<xs:element name="Version" type="xs:string"/>
<xs:element name="Author" type="xs:string"/>
<xs:element name="Date" type="xs:date"/>
<xs:element name="Group" type="xs:string"/>
<xs:element name="Lable" type="xs:string"/>
<xs:element name="Value"/>
<xs:element name="Modal" type="xs:boolean"/>
<xs:element name="Comments">
<xs:complexType>
  <xs:sequence> ... </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Grammar">
<xs:complexType>
 <xs:sequence> ... </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name= "Position">
<xs:complexType>
 <xs:sequence>
     <xs:element ref="Parameters"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="Prm">
 <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="$([a-zA-Z0-9])*[@]([a-zA-Z0-9])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

<xs:element name="Optional">
<xs:complexType>
  <xs:sequence>
    <xs:element name="Prm" type="xs:string" maxOccurs="unbounded" minOccurs="0"/>
  </xs:sequence>
```

Figure 9 - XSD file for develping of part programming specifications

5. Case studies

Using the meta-model of CNC part programming languages, a Universal Process Comprehension interface titled UPCi (Zhang *et al.* 2013b) has been developed as highlighted in Figure 5 in the rectangular box. UPCi is designed to capture the shopfloor knowledge and reconstruct the original process plan from low-level part programs. It is useful to realise the interoperability between CNC machines and other computer systems. The activity flowchart of UPCi is illustrated in Figure 10. There are mainly three modules: loading and translating G&M codes into meta-model data, feature recognition (Zhang *et al.* 2013a) and output generation. The output of UPCi is STEP-NC representation of process plan. The standardised format of the output enables the possibility to be used in other engineering systems.

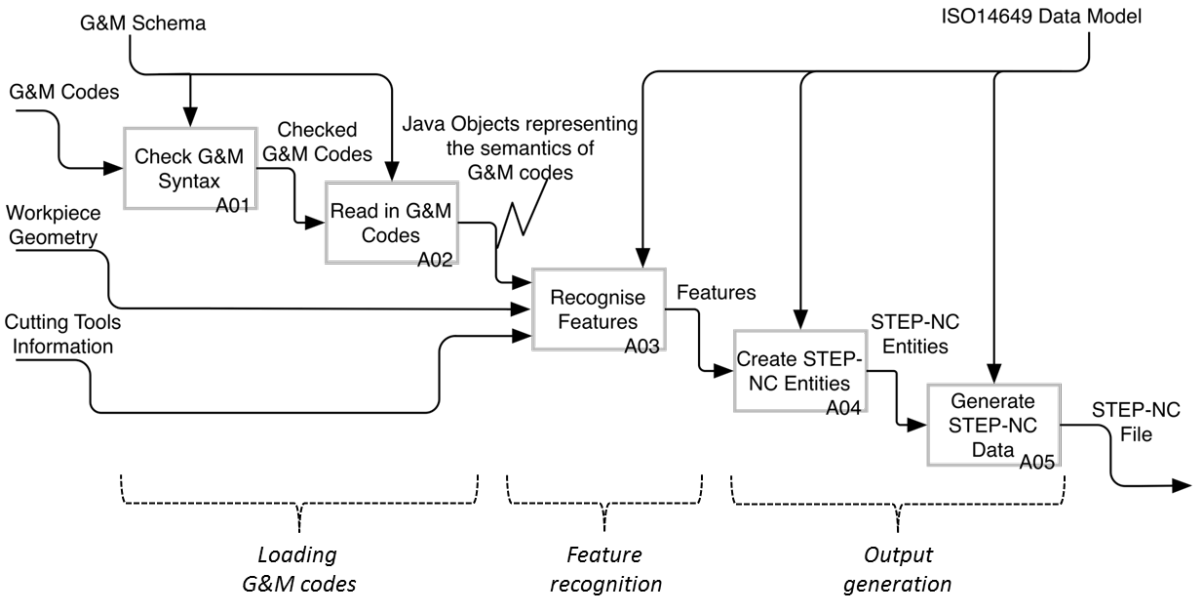


Figure 10 - IDEF0 view of UPCi activities

To validate the efficiency of using the meta-model to deal with different CNC part programming languages, a validation method as shown in Figure 11 has been proposed. In this method, the component is designed in a CAD/CAM system and through two different post processors, two copies of part programs have been generated. Post Processor 1 is for Fanuc 18i controller and Post Processor 2 is for Siemens 840D controller. Though UPCi, two STEP-NC files can be generated. To compare these two STEP-NC files, the semantics in the two file should be identical since they are all derived from the same process plan from CAD/CAM system. The semantic identity between the two STEP-NC file can be a proof of the efficiency of the meta-model. It is worthy to note that, in each case study a Fanuc part programs has been used to machine the part, where the operator has made minor changes (starting point, tool names) to the part programs at the shopfloor.

Two different components, as shown in Figure 12, have been used in this validation method. Figure 13 pictures the generation of STEP-NC data within UPCi. The two STEP-NC files as shown in Figure 14 and Figure15, generated for component 1 have been compared. From the two figures, the process plans extracted from the two pieces of part programs are exactly identical except for the different tools and shopfloor changes to the Fanuc part program. The entities highlighted in dashed rectangular shows that the operations' sequence, the planar feature and facing operation and the cutting parameters/strategies for facing are the same between two STEP-NC files. The different entities (#26, #34, #41) have been highlighted as well in solid-lined rectangular. The full results of the different entities between the two STEP-NC files are listed in the Table 3.

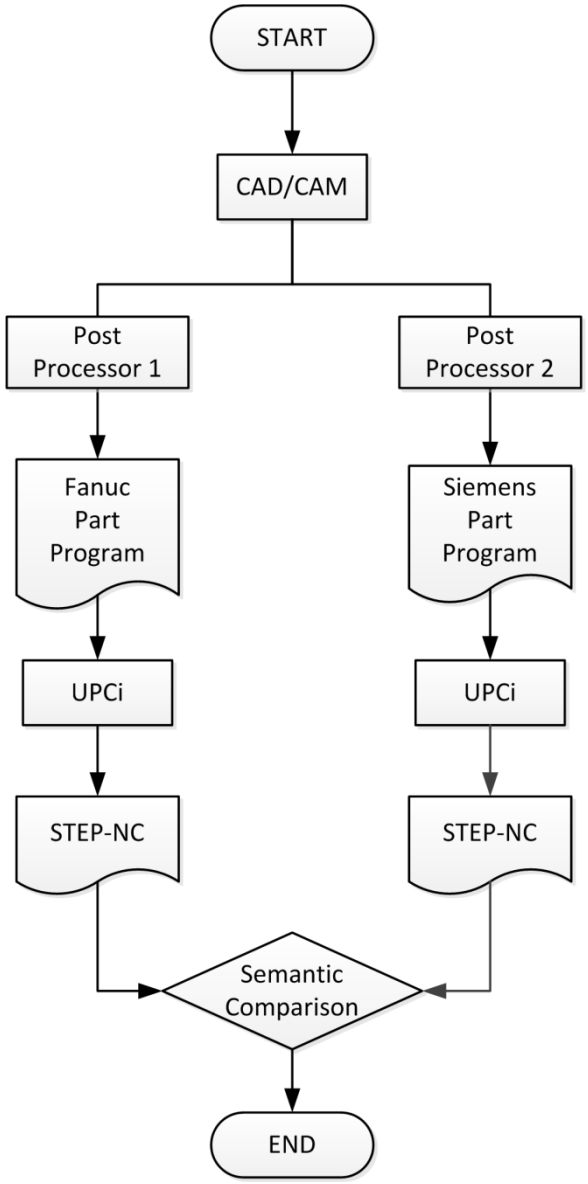
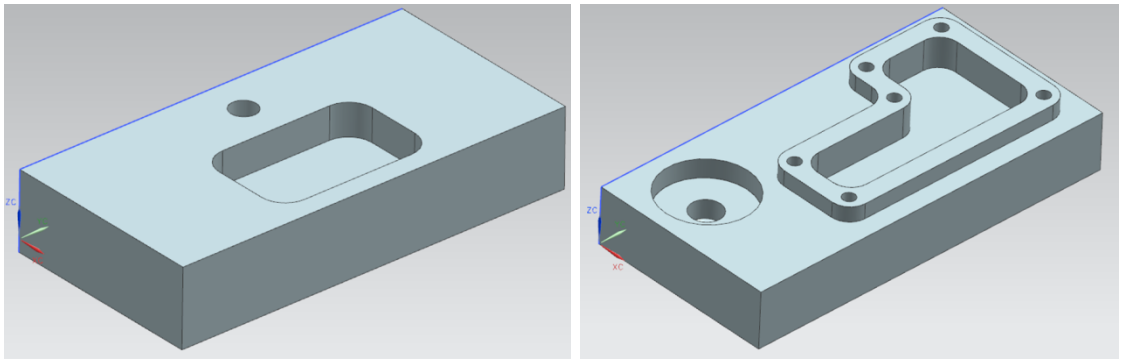


Figure 11 - Validation method of the meta-model



(a) Test Component 1 (b)Test Component 2

Figure 12 - 3D designs of the test components

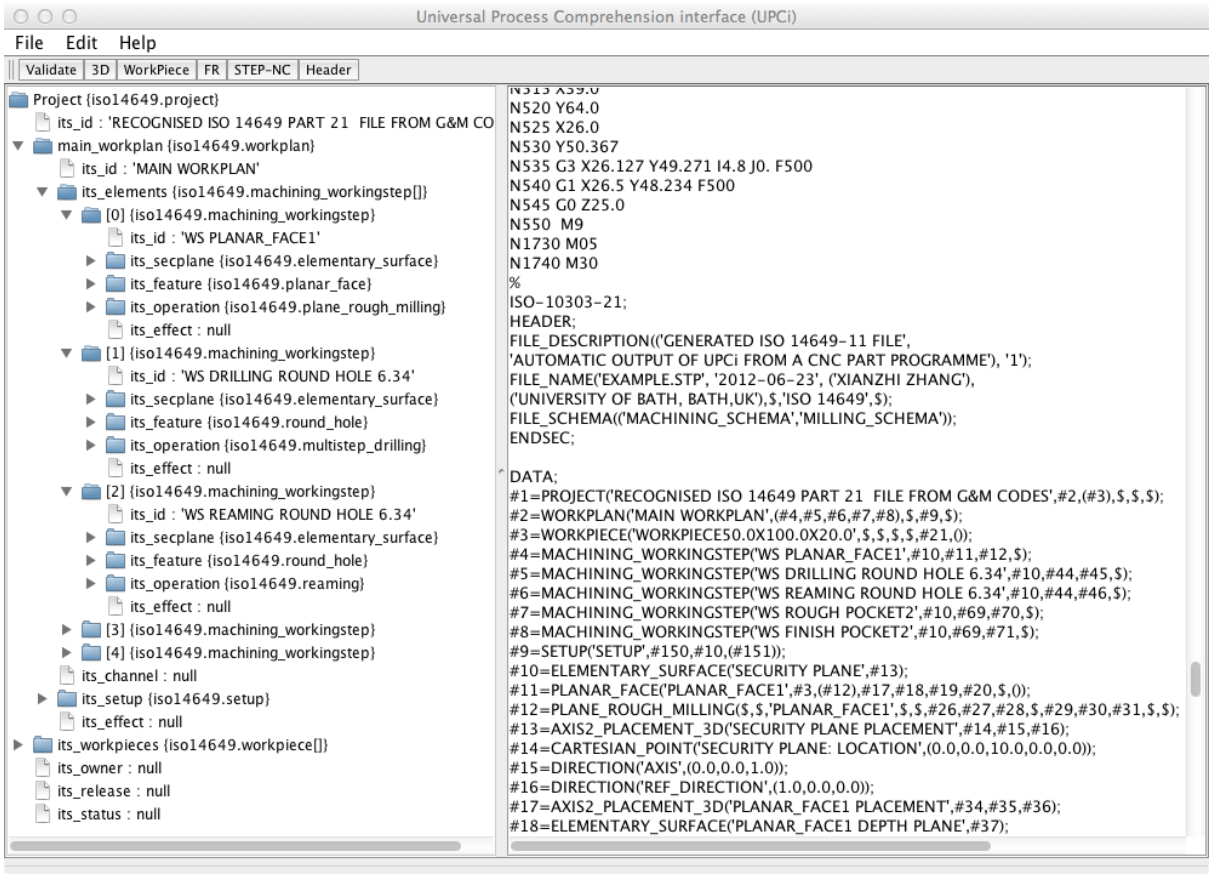


Figure 13 - STEP-NC file generation within UPCi from Fanuc 81i part program for component 1

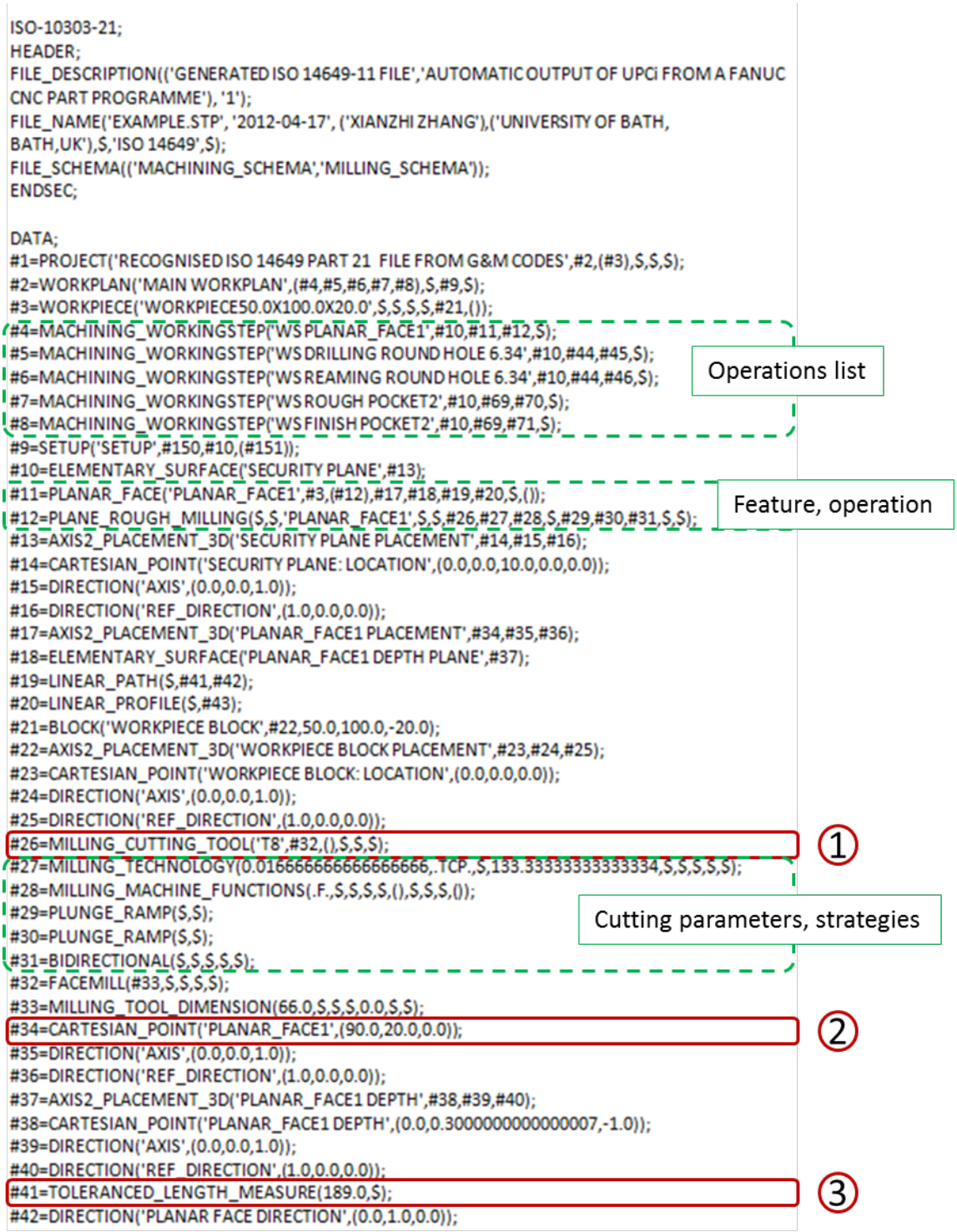


Figure 14 - STEP-NC file from Fanuc 18i part programs for component 1

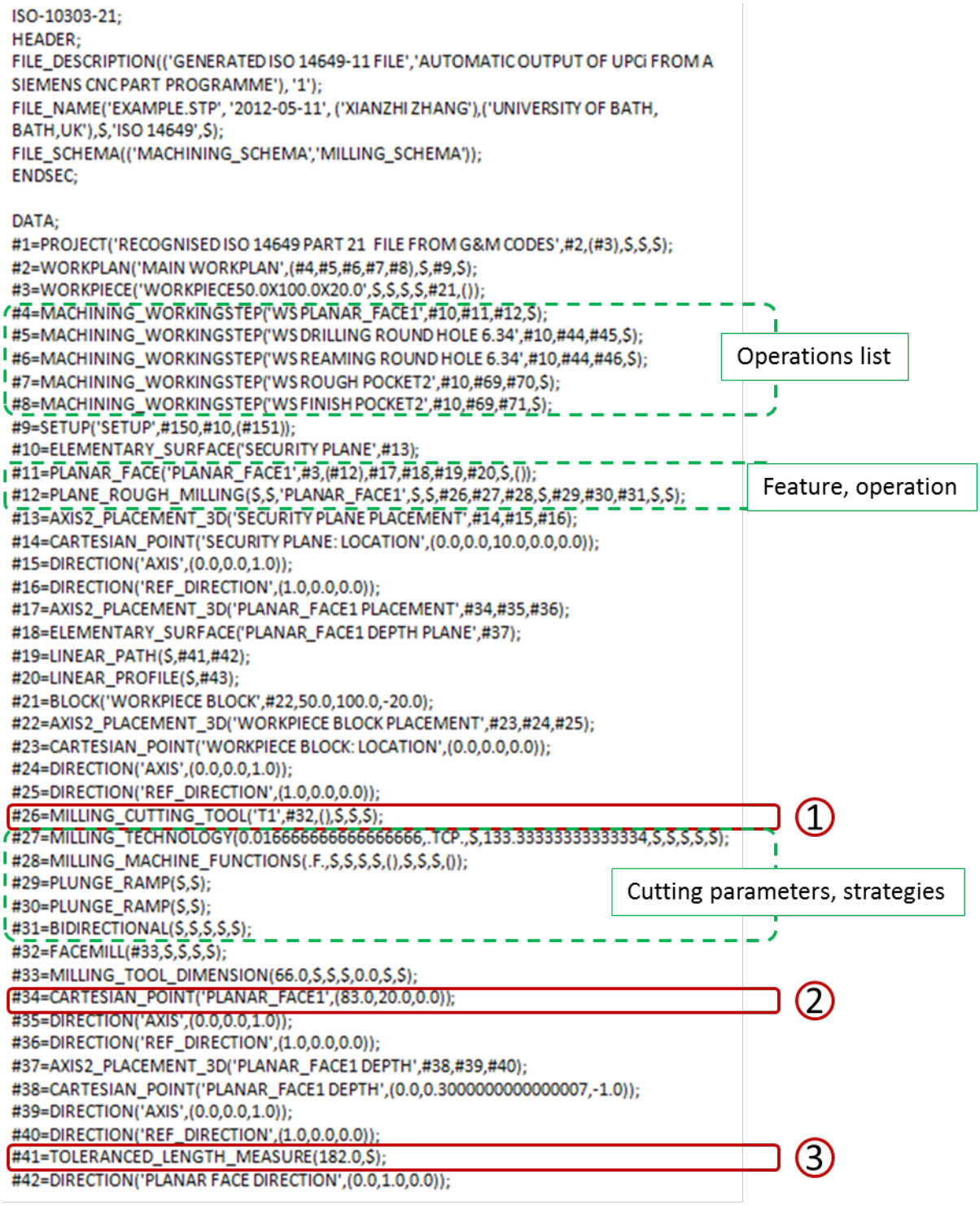


Figure 15 - STEP-NC file from Siemens 840D part programs for component 1

Table 3 - Comparison between two generated STEP-NC files for component 1

1	Fanuc 18i	#26=MILLING_CUTTING_TOOL('T8',#32,(),$,$,$);
	Siemens 840D	#26=MILLING_CUTTING_TOOL('T1',#32,(),$,$,$);
2	Fanuc 18i	#34=CARTESIAN_POINT('PLANAR_FACE1',(90.0,20.0,0.0));
	Siemens 840D	#34=CARTESIAN_POINT('PLANAR_FACE1',(83.0,20.0,0.0));
3	Fanuc 18i	#41=TOLERANCED_LENGTH_MEASURE(189.0,$);
	Siemens 840D	#41=TOLERANCED_LENGTH_MEASURE(182.0,$);
4	Fanuc 18i	#50=MILLING_CUTTING_TOOL('T9',#54,(),$,$,$);
	Siemens 840D	#50=MILLING_CUTTING_TOOL('T2',#54,(),$,$,$);
5	Fanuc 18i	#56=MILLING_CUTTING_TOOL('T7',#60,(),$,$,$);
	Siemens 840D	#56=MILLING_CUTTING_TOOL('T3',#60,(),$,$,$);
6	Fanuc 18i	#77=MILLING_CUTTING_TOOL('T10',#83,(),$,$,$);
	Siemens 840D	#77=MILLING_CUTTING_TOOL('T4',#83,(),$,$,$);

From Table 3, between these two STEP-NC files there are 6 differences, 4 of which are tool names. It is common that tools with the same dimension have different names on two different machines. The other two differences (number 2 and 3) are due to the different starting points to mill the top surface. Actually, the difference starts from shopfloor changes to the Fanuc 18i part program. At the shopfloor it was thought it is a good practice to place the tool a little further from the part and then cut into the material for safety reasons. It is evidence that the meta-model proposed in this paper can be used to capture the shopfloor knowledge from part programs. However, it doesn't make a difference to the features and the machining methods. Hence, the two STEP-NC files generated from Fanuc and Siemens part programs are semantically identical.

Similarly, for the two STEP-NC files generated by UPCi from part programs for component 2, the comparison has been conducted and differences have been listed in Table 4. The results are similar to the first component. Five differences are related to different tool names. Two differences are due to the modification of the starting point of the surface milling operation made at shopfloor. Semantically, the two STEP-NC files are identical.

Table 4 - Comparison between two generated STEP-NC files for component 2

1	Fanuc 18i	#34=MILLING_CUTTING_TOOL('T8',#40,(),$,$,$);
	Siemens 840D	#34=MILLING_CUTTING_TOOL('T1',#40,(),$,$,$);
2	Fanuc 18i	#42=CARTESIAN_POINT('PLANAR_FACE1',(90.0,20.0,0.0));
	Siemens 840D	#42=CARTESIAN_POINT('PLANAR_FACE1',(83.0,20.0,0.0));
3	Fanuc 18i	#49=TOLERANCED_LENGTH_MEASURE(189.0,$);
	Siemens 840D	#49=TOLERANCED_LENGTH_MEASURE(182.0,$);
4	Fanuc 18i	#60=MILLING_CUTTING_TOOL('T5',#66,(),$,$,$);
	Siemens 840D	#60=MILLING_CUTTING_TOOL('T2',#66,(),$,$,$);
5	Fanuc 18i	#176=MILLING_CUTTING_TOOL('T4',#182,(),$,$,$);
	Siemens 840D	#176=MILLING_CUTTING_TOOL('T3',#182,(),$,$,$);
6	Fanuc 18i	#304=MILLING_CUTTING_TOOL('T7',#308,(),$,$,$);
	Siemens 840D	#304=MILLING_CUTTING_TOOL('T4',#308,(),$,$,$);
7	Fanuc 18i	#322=MILLING_CUTTING_TOOL('T22',#326,(),$,$,$);
	Siemens 840D	#322=MILLING_CUTTING_TOOL('T5',#326,(),$,$,$);

6. Conclusions

Due to the existence of many different part programming languages and the mechanism of part program generation, the shopfloor CNC machines have been isolated from other computer aided systems. The shopfloor knowledge cannot be captured and reused automatically. This paper has realised a universal representation of different CNC programming dialects for the purpose of interoperability. A meta-model of CNC programming languages has been proposed. To translate CNC dialects into this meta-model without developing loads of translation interfaces, a dictionary method is used. Programming specifications of CNC dialects have been modelled in XML format to realise the standardised translation of CNC dialects. The meta-model together with the XML description of CNC dialects enables the UPCi to be an expansible system for new programming dialects. The XML schema is used to ensure the standardisation of the XML specifications of NC programming languages and entitle robustness to UPCi. The results of the paper show that the meta-model is a highly effective method to represent and interpret the process plan within part programs, and capture the shopfloor knowledge. The utilisation of STEP-NC as the representation of the process plan makes it neutral and standardised, and it is possible for the process plan to be interpreted and reused by other systems in the CAx chain. It proposes a novel technique to connect the shopfloor with upper stream planning departments and provides new thinking and perspective to capture and reuse the process knowledge from the shopfloor.

References

Bex, G.J., Neven, F. and Van den Bussche, J., 2004. DTDs versus XML schema: a practical study. Proceedings of the Seventh International Workshop on theWeb and Databases (WebDB 2004), Paris, France. pp.79-84. 17-18 June.

GE Fanuc Automation, 1998. *GE Fanuc Automation Operator's Manual, Series 16i/18i/160i/180i - Model A, for Machining Center*. GE Fanuc Automation.

GibbsCAM,2012. Press Releases 2012: GibbsCAM Generates CNC Programs for Nakamura-Tome MTMs [Online]. Available from [http://www.gibbscam.com/news_events/index.php?page=press-releases&con=369].

Guo, X., Liu, Y., Du, D., Yamazaki, K. and Fujishima, M., 2011. A universal NC program processor design and prototype implementation for CNC systems. *The International Journal of Advanced Manufacturing Technology*, pp.1-15.

Hardwick, M. and Loffredo, D., 2006. Lessons learned implementing STEP-NC AP-238. *International Journal of Computer Integrated Manufacturing,* 19(6), pp.523-532.

Heidenhain,2009. HEIDENHAIN TNCguide [Online]. Available from [http://www.heidenhain.com/].

ISO6983-1, 1982. *Numerical control of machines—program format and definition of address words, Part 1: data format for positioning, Line motion and contouring control systems*. International Standards Organisation (ISO).

Lee, D. and Chu, W.W.,2012. Comparative Analysis of Six XML Schema Languages [Online]. Available from [http://www.cobase.cs.ucla.edu/tech-docs/dongwon/sigmod-record-00.html].

Liu, Y., Guo, X., Li, W., Yamazaki, K., Kashihara, K. and Fujishima, M., 2007. An intelligent NC program processor for CNC system of machine tool. *Robotics and Computer-Integrated Manufacturing,* 23(2), pp.160-169.

Maeder, W., Nguyen, V.K., Richard, J. and Stark, J., 2002. Standardisation of the Manufacturing Process: the IMS STEP-NC project. Proceedings of the INational Network of Competence on Integrated Production Logistics Workshop, Saas Fee, Switzerland. pp.5.5/1-5.5/3. Sep. 10-12.

Nassehi, A., 2007. *The realisation of CAD/CAM/CNC interoperability in prismatic part manufacturing*. University of Bath, Bath, UK.

Newman, S.T., Nassehi, A., Xu, X.W., Rosso Jr, R.S.U., Wang, L., Yusof, Y., Ali, L., Liu, R., Zheng, L.Y., Kumar, S., Vichare, P. and Dhokia, V., 2008. Strategic advantages of interoperability for global manufacturing using CNC technology. *Robotics and Computer-Integrated Manufacturing,* 24(6), pp.699-708.

Nguyen, V. and Stark, J., 2005. STEP-compliant CNC Systems, Present and Future Directions, Advanced Design and Manufacturing Based on STEP. Springer.

Proctor, F.M., Kramer, T.R. and Michaloski, J.L., 1997. Canonical machining commands, NISTIR 5970. ISD of NIST, USA.:

Proctor, F.M., Michaloski, J.L. and Shackleford, W.P., 2002. Tying together design, process planning and machining with STEP-NC technology. Proceedings of the 5th Biannual World Automation Congress, Orlando, Fl. pp.33-38. Jun 09-13.

Schroeder, C., 1998. *Printed circuit board design using autoCAD, EDN series for design engineers*. Newnes.

Schroeder, T. and Hoffmann, M., 2006. Flexible automatic converting of NC programs. A cross-compiler for structured text. *International Journal of Production Research,* 44(13), pp.2671-2679.

Siemens AG, 2000. *Short Guide Programming: SINUMERIK 840D/840Di 810D/FM-NC*: Siemens AG.

Smid, P., 2003. *CNC programming handbook : a comprehensive guide to practical CNC programming,* 2nd ed. New York: Industrial Press.

STEP Tools Inc,2012. STEP AP 238 FAQ: Questions about Manufacturing Data Standards [Online]. Available from [http://www.steptools.com/library/stepnc/faq/faq_02.html].

W3C,2012. XML Schema [Online]. Available from [http://www.w3.org/XML/Schema].

Zhang, X., Nassehi, A. and Newman, S., 2013a. Feature recognition from CNC part programs for milling operations. *International Journal of Advanced Manufacturing Technology*.

Zhang, X., Nassehi, A., Safaieh, M. and Newman, S.T., 2013b. Process comprehension for shopfloor manufacturing knowledge reuse. *International Journal of Production Research*, pp.1-15.